METAFONT

(Version[1] 1.7 / 1.25)

Bedienungsanleitung

Lutz Birkhahn

7. September 1989

[1] Die erste Versionsnummer ist die offizielle Zählung von Donald E. Knuth, die zweite bezeichnet die aktuelle Implementation auf dem Atari ST

Dritte Auflage.
September 1989

Inhaltsverzeichnis

Kapitel 1

Einführung

Im fünfzehnten Jahrhundert wurde das erste mal versucht, Buchstaben auf mathematischem Wege zu erzeugen. Nach einer Blütezeit im sechzehnten und siebzehnten Jahrhundert wurde diese Methode im achtzehnten Jahrhundert aufgegeben, die Ergebnisse waren einfach zu schlecht. Erst seitdem in der heutigen Zeit Computer die Berechnungen schnell und exakt durchführen können (auch die Mathematik ist während dieser Zeit nicht stehen geblieben), scheint es möglich und vielleicht auch lohnend zu sein, die Mathematik zur Erzeugung von Buchstaben zu verwenden.

Der bekannte Mathematiker und Informatiker Donald E. Knuth hat sich seit 1977 mit dem weiten Gebiet der Typographie beschäftigt, nachdem er die ersten mit Computerhilfe hergestellten Drucke seiner Buchreihe „The Art of Computer Programming“ gesehen hatte. Die erste Auflage dieser Buchreihe war noch im Bleisatzverfahren hergestellt worden, und Knuth war nach einem Vergleich des damals noch neuen Computersatzes mit den herkömmlichen Druckverfahren so enttäuscht von der neuen Technik, daß er sich seine eigenen Gedanken über Typographie in Verbindung mit Computern machte. Ergebnisse dieser Forschungsarbeiten waren das Schriftsatzprogramm TEX78 und METAFONT79, ein Programm für den Entwurf von Schriften für rasterorientierte Ausgabegeräte mit mathematischen Mitteln. Nach einigen Jahren Erfahrung mit diesen beiden Programmen hat er 1982 eine neue Version von TEX und 1984 ein neues METAFONT geschrieben. METAFONT entwickelte

er praktisch vollkommen neu, nachdem sich gezeigt hatte, daß der bisherige Ansatz erhebliche Schwächen aufwies.

Dieses neue METAFONT (und auch das neue TeX) hat Knuth in der von ihm erdachten, Pascal-ähnlichen Sprache WEB formuliert und in Buchform [2][3] veröffentlicht. Das eigentlich neue an WEB ist die Kombination von Programm und Dokumentation in *einer* Datei, wofür Knuth den Begriff „Literarisches Programmieren“ eingeführt hat[5]. Und in der Tat sind die Programme fast so gut lesbar wie ein Roman, und sie verdeutlichen, was Knuth von der „Kunst, zu programmieren“ versteht. Mein Dank geht an Knuth, daß er seine Kunst nicht nur theoretisch in den Büchern „The Art of Computer Programming“ veröffentlicht, sondern sie auch praktisch in den Programmen TeX und METAFONT vorgeführt hat.

Angesichts dieser großen Leistungen war es für Stefan Lindner und mich – als wir im Frühling 1987 auf diese Programme stießen – klar, daß wir sie für den Atari ST implementieren wollten, und daß die angepaßten Programme als Shareware verbreitet werden sollten. Wegen der Form von WEB-Dateien und der Tatsache, daß wir beide keine großen Freunde von Pascal sind, kam uns beiden sofort der Gedanke, die Programme in die Programmiersprache C zu übersetzen. Also besorgten wir uns die entsprechenden Bücher von Knuth, und tippten sie ab, wobei wir die Programme gleich in C übersetzten. Stefan implementierte TeX, während ich mich auf METAFONT stürzte. Nach über einem Jahr Tippfehlersuche und Ausprobieren verschiedener C-Compiler waren im Sommer 1988 erste brauchbare Ergebnisse zu sehen. Inzwischen hat sich Turbo C von Borland/Heimsoeth als der am besten geeignete Compiler herausgestellt, und es konnten noch einige Verbesserungen an der Benutzeroberfläche von TeX und METAFONT vorgenommen werden. Die vorliegende Version 1.2 von METAFONT und die Version 1.4 von TeX sind das Ergebnis der langen Bemühungen, Knuths Programme für Besitzer von Atari ST Computern zugänglich zu machen. Beide sind TRAP- bzw. TRIP-getestet, haben also nachgewiesen, daß sie sich auch bei „unmöglichen“ Eingaben genau wie die Originale verhalten.

1.1 Der Autor

An dieser Stelle möchte ich mich ganz kurz vorstellen. Geboren bin ich anno 1963 (zufälligerweise erhielt Donald Knuth in diesem Jahr auch seinen Doktortitel in Mathematik vom California Institute of Technology), und entdeckte schon in der Schule mein Interesse für Computer, zunächst an einem Tischrechner von WANG, der noch mit einem richtigen Kernspeicher ausgestattet war, und bei dem man mit Hilfe einer teuren Zusatztastatur sogar Buchstaben (!) eingeben konnte, wenngleich diese auch nur selten richtig im Rechner ankamen. Nach diversen Zwischenschritten (CBM 3000, AIM-65, mein erster eigener Computer, und C-64) wurde ich 1985 schließlich stolzer Besitzer eines Atari ST. Demnächst werde ich wohl das Informatik-Studium beenden müssen, um endlich was „richtiges" zu arbeiten. Inwieweit diese „richtige" Arbeit dem ähnelt, was ich momentan mache, wird sich noch rausstellen. Es ist aber immerhin steigender Bedarf an Leuten zu erkennen, die sich mit TEX, METAFONT und anderen leistungsfähigen Programmen auskennen.

Meine derzeitige Adresse ist:

Lutz Birkhahn
Fürther Str. 6
8501 Cadolzburg 2
BRD

Telefon: 0 91 03 / 28 86

Wer seine Registrierungsgebühr zahlen möchte (Spenden werden natürlich auch angenommen), überweist den Betrag am besten auf folgendes Konto:

Kontonr. 533 45 37 bei den Vereinigten Sparkassen im Landkreis Fürth, BLZ 762 501 10

Wer einen Akustikkoppler oder ein Modem hat, kann mich auch in folgenden Mailboxen erreichen:

- Aided Messages Equipment (AME), Tel.-Nr. 0 91 31 / 99 29 98, Benutzername LUTZ (Parameter: 1200–2400/8/N/1, 24 h online). Diese Mailbox ist an das Zerberus-Netz angeschlossen und somit aus der gesamten Bundesrepublik sowie (bis jetzt) aus der Schweiz und aus Österreich einigermaßen kostengünstig zu erreichen (als Empfänger muß man dann LUTZ@AME.ZER eingeben).

- Mailbox System Nürnberg (MSN), Tel.-Nr. 09 11 / 33 00 39 oder 33 10 40 (Parameter 300/8/N/1) oder 09 11 / 3 73 89 (1200–2400/8/N/1) oder DATEX-P (+262) 45 911 010 290. Bei Name 'ZCZC' und bei Empfänger 'L.Birkhahn' eingeben. Damit hat man einen kostenlosen Zugang zu diesem System und kann mir einen bis zu 4000 Zeichen langen Text senden.

In beiden Mailboxen gibt es TEX-Bretter, in denen aktuelle Informationen, Programme sowie allgemein interessierende Fragen und Antworten zu TEX und METAFONT stehen.

Von Stefan Lindner kann man zu ähnlichen Konditionen wie bei METAFONT das berühmte Schriftsatzprogramm TEX (ebenfalls von Donald Knuth entwickelt) zusammen mit DVI-Gerätetreibern für die gebräuchlichsten Drucker bekommen. Seine Adresse lautet:

Stefan Lindner
Iltisstraße 3
8510 Fürth

Kapitel 2

Shareware

2.1 Shareware-Bedingungen

Die Programmdiskette (Disk 1) darf (und soll) unter den folgenden Bedingungen beliebig kopiert und weitergegeben werden:

- Es werden *alle* Dateien auf der Diskette kopiert.
- Diese Anleitung und die Programme METAFONT und INIMF werden nicht verändert.
- Die Weitergabe erfolgt ausschließlich zu nichtkommerziellen Zwecken (spezielle Bedingungen bitte bei mir erfragen).

Kurz gesagt, ich möchte nicht, daß unvollständige Versionen kursieren, oder daß jemand mit diesen Programmen Geld verdient.

Disk 2 ist ohne Einschränkungen frei kopierbar, ich empfehle jedoch, beide Disketten nur im Paket zu kopieren, denn sie gehören schon sehr eng zusammen.

Ich kann natürlich für die gesamten Programme und die Daten keine Garantie geben, und auch keine eventuell auftretenden Schäden ersetzen (wenn

also z.B. die Augen nicht mehr mitmachen, weil sie durch einen falschen Zeichensatz zu Tode erschreckt sind, dann bitte die Arztrechnung nicht an mich schicken). Ich habe jedenfalls versucht, alles so gut und richtig wie möglich zu machen.

Jeder, dem diese Implementation von METAFONT gefällt, und der die Programme öfter verwendet, wird gebeten, 50,- DM (außerhalb Deutschlands wegen des höheren Portos 55,- DM oder US-$ 30) an meine auf Seite 5 angegebene Adresse zu schicken. Damit ist er ein registrierter Benutzer und erhält folgenden Service:

- Die neueste Version (mit Seriennummer, dazu unten mehr) wird kostenlos zugeschickt, zusammen mit dieser Bedienungsanleitung in gedruckter Form.
- Telefon-Hotline: kostenlose Beratung registrierter Benutzer[1].
- Registrierte Benutzer können für 30,- DM (oder US-$ 20) den kompletten Quellcode (in C) zu METAFONT anfordern.
- Das erste Update wird kostenlos zugeschickt (wo kriegt man heute noch so einen Service?), alle weiteren Updates werden schriftlich bekanntgegeben, registrierte Benutzer erhalten diese dann gegen voraussichtlich ca. 10,- DM pro Diskette.

2.2 Seriennummer

In jeder METAFONT-Version ist eine Seriennummer enthalten. Wenn jemand (im folgenden A genannt) den oben genannten Betrag bezahlt, erhält er dafür eine Version mit einer neuen Seriennummer. Diese Nummer wird bei mir registriert. Wenn er nun diese registrierte Version weitergibt (und das soll er

[1] Das soll natürlich nicht heißen, daß nur registrierte Benutzer bei mir anrufen dürfen. Über Lob, Kritik, Verbesserungsvorschläge oder gar selbstgeschriebene Programme zu METAFONT oder Ideen dazu freue ich mich immer. Nur kann es passieren, daß ich bei Fragen, zu deren Beantwortung ich selbst erstmal etwas Zeit investieren muß, zunächst frage, ob der Anrufer registriert ist.

ja), und einer der Empfänger dieser Kopie zahlt seinerseits die Sharewaregebühr, so erhält A davon 15,- DM (bzw. ca. US $ 8) als „Prämie“ für die Verbreitung des Programmes. Wenn also nur drei von den Leuten, an die man die Diskette weitergegeben hat, sich bei mir registrieren lassen, hat man bereits seine Ausgaben fast amortisiert.

Wenn jemand allerdings mehr als dreißigmal die Belohnung kassiert, gehe ich davon aus, daß es sich um einen kommerziellen Software-Vertrieb handelt, und gebe an diesen keine Belohnungen mehr weiter.

Mit diesem Konzept will ich einerseits die Verbreitung von METAFONT etwas beschleunigen, und andererseits versuchen, mögliche Anfangsschwierigkeiten durch persönlichen Kontakt unter den Benutzern zu verringern (damit ein Benutzer, der weit entfernt von mir wohnt, bei Problemen mit METAFONT zunächst zum vermutlich näher gelegenen Bekannten gehen kann, von dem er die Kopie hat, bevor er bei mir anruft, um das Problem zu lösen. Vieles läßt sich eben bei einem persönlichen Gespräch viel leichter und billiger lösen als über eine lange Telefonleitung).

Damit ich die „Prämie“ weiterleiten kann, bitte bei der Registrierung unbedingt die Seriennummer der bisher benutzten Version (erscheint bei jedem Start von METAFONT in der zweiten Zeile auf dem Bildschirm) mit angeben. Falls man bisher noch keine Version besaß (ist mir im Moment allerdings noch unklar, wie man dann an diese Anleitung kommen kann), sollte man dies bitte deutlich vermerken. Am besten druckt man die Datei `FORMULAR.TXT` auf einem Drucker aus und macht die entsprechenden Eintragungen oder sendet das ausgefüllte Formular an mich in eine der auf Seite 5 angebenen Mailboxen.

Kapitel 3

Installation

Alle Hinweise in diesem Kapitel sind lediglich Vorschläge für Benutzer, die bis jetzt noch wenig Erfahrung mit METAFONT haben. Durch die Verwendung einer „Setup-Datei“ kann man die Programme und Daten fast beliebig auf seinen Speichermedien verteilen.

Da die kompletten Sourcecodes der Zeichensätze zusammengenommen mehr als 700 KByte Umfang besitzen, war es nötig, die Dateien aus dem Ordner `METAFONT\MFINPUTS\CMR` mit dem Programm ARC zu komprimieren. Dieses Programm und eine Shell mit GEM-Oberfläche befinden sich ebenfalls auf Diskette 2. Bezüglich der Weitergabe dieser Programme verweise ich auf die jeweiligen Anleitungen. Ein weiterer Vorteil der Komprimierung mit ARC besteht darin, daß für sämtliche Dateien eine Prüfsumme gebildet wird. Wenn ARC beim Expandieren der Daten (dazu den Button [`EXTRACT from Arc`] anklicken) keine Fehler meldet, ist somit sichergestellt, daß alle Daten vollständig und fehlerfrei sind.

3.1 Installation auf Festplatte

Besitzer einer Festplatte sind (nicht nur) bei der Installation fein raus: Sie starten einfach das Programm `INSTALL.PRG` auf Diskette 1, tragen in der daraufhin erscheinenden Dialogbox den gewünschten Pfad für METAFONT

in der entsprechenden Zeile ein, selektieren den Button METAFONT, klicken den OK-Button an und befolgen die Anweisungen, die das Programm gibt. Wenn das Programm fertig ist, steht METAFONT gebrauchsfertig auf der Platte, die Setup-Datei ist auch bereits an die örtlichen Gegebenheiten angepaßt. Lediglich bei der TEX-Shell müssen noch die Pfade eingestellt werden (siehe dazu die Anleitung der TEX-Shell). Ein weiterer Vorteil des Installationsprogrammes ist die automatische Überprüfung der Dateien mittels CRC-Prüfsummen.

3.2 Installation auf Disketten-Laufwerk

Für eigene Experimente mit METAFONT, bei denen die CMR-Sourcecodes nicht benötigt werden, kann die Programmdiskette (am besten natürlich eine Sicherheitskopie davon) ohne spezielle Installation verwendet werden. Benutzer, die nur einseitige Laufwerke besitzen, können sich (bei einem Freund mit zweiseitigem Laufwerk) eine Arbeitsdiskette zusammenstellen, auf der nur die Dateien METAFONT.PRG, MFSETUP und PLAIN.BSE stehen. Es bleiben dann noch etwas mehr als 100 KByte übrig, in denen Eingabedateien (*.MF) sowie LOG-, GF- und TFM-Dateien untergebracht werden können. Routinierte Diskjockeys oder Besitzer von zwei Laufwerken können in der Setup-Datei auch Laufwerk B: angeben (z.B. bei den inputpaths). Wer noch etwas Platz im RAM hat, kann eine kleine Ramdisk (Minimum ca. 50 KByte) installieren, und in der Setup-Datei logpath, gfpath oder tfmpath (auf diese Dateien wird in der Regel während des gesamten Programmlaufes etwas geschrieben) auf die Ramdisk setzen.

Um komplette Zeichensätze für TEX zu erzeugen, muß man schon etwas mehr mit dem Diskettenplatz jonglieren. Benutzer mit nur *einem einseitigen* Laufwerk und nicht mehr als 1 MByte RAM müssen hier einen ziemlich hohen Aufwand treiben, es müssen nämlich für jeden zu erzeugenden Zeichensatz erst die entsprechenden Dateien zusammengesucht werden, da die kompletten Sourcen plus METAFONT plus Base-Datei nicht auf zwei einseitige Disketten passen. Um das Zusammensuchen etwas zu erleichtern, folgt ein kurzer Überblick über die Organisation der CMR-Dateien.

Für jeden Zeichensatz wird zunächst eine *Parameterdatei* geladen. Das sind die Dateien, deren Name mit einer Zahl endet (z.B. CMR10.MF und

CMSL12.MF). In jeder Parameterdatei wird am Anfang die Datei CMBASE.MF geladen, in der von allen CMR-Zeichensätzen verwendete Makros und Deklarationen stehen. Am Ende einer Parameterdatei wird mittels des generate-Makros die entsprechende *Treiberdatei* (z.B. ROMAN.MF, ITALIC.MF) geladen, die nun ihrerseits die verschiedenen *Programmdateien* mit den Programmen für die einzelnen Zeichen (z.B. ROMANU.MF, ITALL.MF) lädt. Um zum Beispiel den Zeichensatz CMR10 zu erzeugen, benötigt man:

- CMBASE.MF mit den allgemeinen CMR-Makros
- die Parameterdatei CMR10.MF
- die Treiberdatei ROMAN.MF
- die Programmdateien ROMANU.MF, ROMANL.MF, GREEKU.MF, ROMAND.MF, ROMANP.MF, ROMSPL.MF, ROMSPU.MF, PUNCT.MF, ACCENT.MF sowie die Programme für die Ligatur-Zeichen ROMLIG.MF und COMLIG.MF, da die Variable *ligs* in CMR10.MF auf den Wert 2 gesetzt wurde.

Diese aufwendige Sucherei bleibt jedoch Anwendern mit etwas mehr Speicher erspart. Insgesamt braucht man ca. 1 MByte Speicher auf Disketten und evtl. Ramdisk, um das Programm, die Base-Datei und alle Zeichensatz-Sourcen zu speichern. Es würde hier wohl zu weit führen, für jede mögliche Rechnerkonfiguration (davon gibt es schließlich sehr viele verschiedene) die beste Aufteilung anzugeben, schon allein deswegen, weil es auch auf den Anwendungsfall ankommt, was man als *beste* Aufteilung ansehen kann. Ich denke jedoch, daß es mit diesen Informationen jedem Benutzer möglich sein sollte, eine für seine Anwendung passende Verteilung der Daten zu finden (wenn nicht, gibt es ja immer noch die „Telefon-Hotline“ oder andere METAFONT-Benutzer, die solche Arbeiten schon hinter sich haben).

Kapitel 4

METAFONT auf dem Atari ST

METAFONT wurde auf dem ST als GEM-Programm realisiert. Sämtliche Textausgaben gehen in ein Text-Fenster, dort werden auch die Eingaben des Benutzers vorgenommen. Ein zweites Fenster dient der Ausgabe von Grafik. Dieses wird aber erst auf Anforderung geöffnet, z.B. mit den METAFONT-Befehlen `openit` oder `showit`. Wenn man bei der Erzeugung eines Zeichensatzes jedes Zeichen sehen will, bevor es in die GF-Datei geschrieben wird, kann man dies mit dem METAFONT-Befehl `screenchars` einschalten. Mit `screenstrokes` kann man jeden einzelnen „Pinselstrich" auf dem Bildschirm sehen. Für die weiteren Grafikbefehle sei auf das METAFONTbook [1] verwiesen.

Man kann die Fenster jederzeit vergrößern, verkleinern oder verschieben, auch während METAFONT beschäftigt ist. Bei längeren Dateioperationen (z.B. Laden der Base-Datei) oder komplizierten Berechnungen kann es jedoch vorkommen, daß METAFONT nicht sofort auf die Aktivitäten des Benutzers reagiert, sodaß man entsprechend länger auf die Maustaste drücken muß.

Die Arbeit von METAFONT kann man durch gleichzeitiges Drücken der Tasten CONTROL und ALTERNATE unterbrechen, man landet dann in der normalen Fehlerbehandlungs-Routine, wo man z.B. Endlosschleifen untersuchen kann, Variablen abfragen oder das Programm beenden kann. Aus den gleichen Gründen wie im vorigen Absatz muß man die beiden Tasten eventuell etwas länger gedrückt halten. METAFONT unterbricht nämlich nur

dann seine Arbeit, wenn es auch in der Lage ist, zusätzliche Eingaben des Benutzers zu verdauen. Trotzdem ist es zum Beispiel bei der Untersuchung von Variablen ratsam, die Aktivitäten vor METAFONT zu „verstecken“, damit man nichts durcheinanderbringt. Dies kann man durch das `hide`-Makro erreichen. Um z.B. den Inhalt der Variablen x anzuzeigen, gibt man folgendes ein:

```
I hide(show x)
```

Während METAFONT auf eine Eingabe wartet, kann man es durch Drücken der ESCAPE-Taste abbrechen. Wenn man in der daraufhin erscheinenden Alarm-Box auf `abort` klickt, riskiert man allerdings unvollständige GF- und TFM-Dateien!

4.1 Setup-Datei

In der Setup-Datei stehen sämtliche Pfade, auf die METAFONT zugreifen kann. Wenn in der Kommandozeile nichts anderes angegeben wurde, sucht METAFONT nach der Datei MFSETUP im aktuellen Ordner. Die Syntax für die Pfad-Definitionen ist bis auf die Bezeichnung der Schlüsselwörter genau die gleiche wie beim TEX von Stefan Lindner. Es sind also zwischen den einzelnen Wörtern (Token) beliebig viele Leerzeichen, Returns, Tabulatoren und Kommentare erlaubt. Kommentare werden mit einem ‘`%`’ eingeleitet und gehen bis zum Zeilenende (wie bei TEX und METAFONT üblich). Auf der linken Seite vom ‘`=`’ dürfen folgende Schlüsselwörter stehen (in Klammern ist jeweils angegeben, welchen Pfad METAFONT verwendet, wenn in der Setup-Datei nichts definiert ist). Wenn nichts anderes angegeben ist, gelten die Pfade sowohl für METAFONT als auch für INIMF.

`poolfile` gibt an, von wo INIMF den String-Pool lädt. Dies muß ein gültiger Dateiname sein, er darf also insbesondere nicht mit einem ‘\’ enden, und die Extension muß bei Bedarf explizit angegeben werden. METAFONT ignoriert diesen Eintrag. (`\metafont\mfbases\mf_pool`)

`defaultbase` ist die Base-Datei, die METAFONT lädt, wenn der Benutzer nichts anderes angegeben hat. Auch dies muß wie `poolfile` ein gültiger Dateiname sein. (`\metafont\mfbases\plain.bse`)

`basepaths` dort wird die Base-Datei gesucht (wenn eine angegeben wurde, denn sonst lädt INIMF gar keine, METAFONT die unter `defaultbase` genannte). (`\metafont\mfbases\`)

`inputpaths` besteht in der Regel aus einer Liste von Pfaden, in denen nach Input-Dateien (`*.MF`) gesucht werden soll. (`\metafont\mfinputs\`)

`gfpath` in diesen Ordner werden die GF-Dateien geschrieben. (`\metafont\`)

`tfmpath` ist der Ort, wo die TFM-Datei abgelegt wird. (`\metafont\`)

`logpath` dorthin kommt die LOG-Datei. (`\metafont\`)

`dumppath` ist der Ordner, in den INIMF die Base-Datei schreibt, wenn 'dump' befohlen wurde. (`\metafont\`)

Auf der rechten Seite des `'='` steht jeweils der Pfad, in dem nach der entsprechenden Datei gesucht wird. Dieser Pfad sollte in der Regel absolut angegeben werden, damit METAFONT die Dateien auch dann findet, wenn es von einem anderen Directory oder Laufwerk aus gestartet wurde. Ein Punkt (`'.'`) ist auch ein gültiger Pfad, er bezeichnet das aktuelle Directory. Wenn dies nicht explizit in der Setup-Datei angegeben ist, sucht METAFONT *nicht* automatisch im aktuellen Directory nach einer Datei. Mit Ausnahme von `poolfile` und `defaultbase` (dies sind ja Dateinamen und keine Pfade) dürfen alle Pfade wahlweise mit oder ohne '\' am Ende angegeben werden. Bei `basepaths` und `inputpaths` dürfen mehrere Pfade, jeweils getrennt durch ein Komma, definiert werden. METAFONT durchsucht dann in der vorgegebenen Reihenfolge alle Pfade, bis es die Datei gefunden hat. Jede Pfad-Definition muß mit einem Strichpunkt abgeschlossen werden. Zwischen Groß- und Kleinschreibung wird nicht unterschieden.

4.2 Kommandozeile

METAFONT wurde so implementiert, daß es als GEM-Programm vom Desktop aus möglichst einfach gestartet werden kann. Viele Benutzer verwenden jedoch lieber eine Shell, die ihnen manche Tipparbeit ersparen kann. Als erstes sei hier die hervorragende TEX-Shell von Klaus Heidrich und Reinhard Maluschka genannt, die dem Paket ebenfalls beiliegt. Mancher bevorzugt aber auch textorientierte Shells, die meist an UNIX oder MS-DOS angelehnt sind. Um die Verwendung solcher Shells (sowohl GEM- als auch textorientierte) zu ermöglichen, wurden 3 Optionen definiert, die in der Kommandozeile angegeben werden können. Parameter (*Dateiname* bzw. *Zahl*) dürfen durch ein Leerzeichen von der Option getrennt werden. Verschiedene Optionen *müssen* durch ein Leerzeichen getrennt werden.

-j Wenn METAFONT seine Arbeit beendet hat, wird normalerweise ein akustisches Weck-Signal (sogenanntes „BING“) ausgegeben und gewartet, bis der Benutzer durch einen Tastendruck signalisiert, daß er wieder aufgewacht ist und alle wichtigen und unwichtigen Mitteilungen auf dem Bildschirm gelesen hat. Wenn man jedoch mit der TEX-Shell oder einer Batchdatei z.B. während der Nacht gleich mehrere Zeichensätze hintereinander erzeugen will, wäre es unpraktisch, wenn man alle zehn Minuten zum Computer rennen müsste, nur um auf eine Taste zu drücken. Das kann der Computer schließlich auch selber machen. Mit der Option `-j` kann man METAFONT mitteilen, daß es die Nacht- (oder Tag-) Ruhe nicht durch Gebimmel unterbrechen soll, und daß es am Ende nicht auf eine Taste warten soll, da sowieso niemand die Kommentare auf dem Bildschirm lesen will.

-s *Dateiname* METAFONT soll die Pfaddefinitionen aus der Datei *Dateiname* lesen. Ohne Angabe dieser Option sucht METAFONT nach der Datei `MFSETUP` im aktuellen Directory. Wenn eine Setup-Datei nicht gefunden wird, dann werden die im vorigen Abschnitt angegebenen Standardpfade verwendet. Bei Verwendung der TEX-Shell in der momentanen Version empfehle ich jedoch, immer die Datei `MFSETUP` zu verwenden, und dort auch alle Pfade zu definieren, sonst kann es zu Problemen im Zusammenspiel der verschiedenen Programme kommen (in

zukünftigen Versionen der TEX-Shell wird es vielleicht ein Telepathie-Modul geben, das die Gedanken des Benutzers liest und daraus die gewünschten Pfade ableitet...).

-e *Zahl* Bei Verwendung der TEX-Shell ist es durch diese Option möglich, direkt aus der Fehlerbehandlung in METAFONT durch Eingabe von 'e' einen Editor zu starten, der sofort die fehlerhafte Datei lädt und in die entsprechende Zeile springt (sofern der Editor ähnlich wie Tempus Dateiname und Zeilennummer in der Kommandozeile akzeptiert). Wie funktioniert nun das Ganze? METAFONT interpretiert die *Zahl* als Adresse im Hauptspeicher und erwartet dort einen Speicherbereich von mindestens 270 Zeichen, der mit dem nullterminierten String 'TEXSHELL' vorbesetzt ist. Wenn der Benutzer nun eine Fehlermeldung mit 'e' beantwortet, schreibt METAFONT den Namen der gerade bearbeiteten Datei und durch ein Leerzeichen davon getrennt die aktuelle Zeilennummer in diesen Speicherbereich (dabei wird die Zeichenfolge "TEXSHELL" überschrieben). Diese Informationen kann das aufrufende Programm auswerten und anschließend den Editor starten.

Wenn man eine Option eingibt, die METAFONT nicht kennt, wird eine Hilfsseite („Usage") ausgegeben, in der nochmal die genaue Verwendung der Optionen erklärt ist.

Zusätzlich zu den eben beschriebenen Optionen kann man bereits in der Kommandozeile eine Startzeile eingeben. Doch dazu erstmal eine Erklärung, was mit Startzeile gemeint ist. Wenn METAFONT vom Desktop aus gestartet wird, meldet es sich mit zwei Sternen ('**'). Damit läßt es erkennen, daß es auf die Eingabe der Startzeile wartet (die normale Eingabezeile beginnt mit einem einfachen Stern). Die Besonderheiten, die für die Startzeile gelten, werden im Kapitel 6 näher erläutert. Diese Startzeile kann man nun bereits in der Kommandozeile angeben, und zwar *nach* den Optionen. Es wäre zwar sehr ungewöhnlich, aber falls die Startzeile zufällig mit einem Minuszeichen ('-') beginnen sollte, muß man diese erstmal durch zwei Minuszeichen von den Optionen abtrennen. Eine Kommandozeile sieht also ganz allgemein so aus:

METAFONT.PRG [*Optionen*] [--] [*Startzeile*]

Falls man die Startzeile bereits in der Kommandozeile angibt, sollte man beachten, daß manche Shells zunächst alle Buchstaben in Großbuchstaben umwandeln, während die Kommandos in METAFONT in der Regel klein geschrieben sind (die bekannteste Shell mit diesem Verhalten ist wohl das GEM-Desktop, das allerdings für GEM-Programme normalerweise keine Kommandozeile vorsieht). Also entweder eine „vernünftige" Shell verwenden oder die Startzeile erst in METAFONT (nach den zwei Sternen) eingeben.

Da in METAFONT der Backslash ('\') als Sonderzeichen definiert ist, darf er in Dateinamen in der Regel nicht verwendet werden (Ausnahmen: wenn METAFONT den Benutzer explizit zur Eingabe eines Dateinamens auffordert sowie bei der '-s'-Option). Um Dateien trotzdem mit einem Pfad versehen zu können, wurde der einfache Schrägstrich ('/') als Ersatzzeichen gewählt (man ist damit auch dem Filesystem von UNIX ein kleines Stück näher). Vor dem Öffnen einer Datei wird dieses Zeichen wieder in einen Backslash zurückverwandelt, damit GEMDOS korrekt arbeiten kann. Generell läßt sich dazu aber sagen, daß man zumindest in den Eingabedateien auf Pfadangaben möglichst verzichten sollte, denn portabel ist sowas natürlich nicht. Schon der Nachbar hat auf seinem Atari ST mit sehr großer Wahrscheinlichkeit eine andere Directory-Struktur, ganz zu schweigen von anderen Computersystemen, wo Pfade möglicherweise ganz anders aufgebaut sind.

4.3 Rückgabewerte von METAFONT

Jedes Programm gibt an das aufrufende Programm eine Zahl zurück, den Rückgabewert. Mit Hilfe dieses Wertes kann man mit den meisten Shells eventuell aufgetretene Fehler oder besondere Ereignisse während des Programmlaufes erkennen und entsprechend darauf reagieren. Es ist üblich, daß bei einem fehlerfreien Lauf die Zahl Null zurückgegeben wird. Da bei der Erzeugung eines Zeichensatzes eine ganze Menge ungewöhnliches passieren kann, wurden für METAFONT sechs weitere Werte definiert (übrigens gelten genau die gleichen Konventionen auch für das TEX von Stefan Lindner). Im folgenden also eine vollständige Liste der Werte, die METAFONT an das aufrufende Programm zurückgeben kann:

0 – no error Es lief alles zur vollsten Zufriedenheit von METAFONT.

1 – warning Während des Programmlaufes wurde mindestens eine Warnung ausgegeben, es wurden jedoch keine ernsthaften Fehler gefunden.

2 – error Mindestens ein Fehler ist aufgetreten, die Arbeit wurde aber korrekt beendet, die erzeugten Daten sind (eventuell beschränkt) verwendbar.

3 – fatal error METAFONT mußte wegen eines fatalen Fehlers abgebrochen werden, die erzeugten Dateien sind höchstwahrscheinlich unvollständig. Dies kann z.B. passieren, wenn METAFONT keine oder nur eine fehlerhafte Base-Datei finden konnte, wenn mehr als 100 normale Fehler aufgetreten sind, ein interner Fehler[1] entdeckt wurde oder der interne Speicher (siehe Speicherverwaltung) übergelaufen ist.

4 – edit Der Benutzer hat bei einem aufgetretenen Fehler oder einer Unterbrechung (Interrupt) ein 'e' eingetippt. Wenn mittels der '-e'-Option ein Speicherbereich spezifiziert wurde, schreibt METAFONT in diesen den Namen der aktuellen Eingabedatei und die Zeilennummer.

5 – exit Wenn der Benutzer in der Fehlerbehandlung ein 'x' eintippt oder während einer Eingabe die Escape-Taste betätigt, wird dieser Wert zurückgeliefert.

6 – low memory Zeigt an, daß der freie Hauptspeicher (RAM) nicht ausreicht, um METAFONT starten zu können (siehe Speicherverwaltung). Abhilfe schafft nur entweder das Entfernen nicht benötigter residenter Programme oder der Kauf von mehr Speicher.

4.4 Speicherverwaltung

Beim Programmstart fordert METAFONT einen konstanten Speicherbereich („interner Speicher“) vom Betriebssystem an, in den es alle Variablen speichern kann. Dieser Speicher ist in verschiedene Bereiche aufgeteilt (z.B. für Strings, Eingabepuffer, verschiedene Kernings, „main memory“ für Zahlen,

[1] ist zwar unwahrscheinlich, aber nie völlig auszuschließen! In diesem Fall bitte ich um sofortige Benachrichtigung zusammen mit einer möglichst genauen Fehlerbeschreibung

Pfade, Pens, Transformationsmatrizen usw.). Wenn das Betriebssystem diesen Speicher nicht zur Verfügung stellen kann (wegen residenter Programme, RAM-Disk o.ä.), wird der Rückgabewert 6 zurückgeliefert. Wenn METAFONT jedoch während der Arbeit feststellt, daß einer der Bereiche im internen Speicher zu klein ist (z.B. bei endlos rekursiven Makros, zu vielen Strings, zu langen Zeilen, ...), so meldet es „`capacity exceeded`" zusammen mit der Angabe, welcher Bereich gesprengt wurde. In diesem Fall kann der Benutzer nur versuchen, sparsamer mit den Ressourcen umzugehen (das sagt sich natürlich leicht, aber es ist kaum möglich, hier konkretere Hinweise zu geben; ein paar Tips kann man noch im METAFONTbook[1] von Knuth finden). Im äußersten Notfall kann man auch bei mir anfragen, ob es möglich ist, den entsprechenden Bereich zu vergrößern; das hätte jedoch auf jeden Fall eine Änderung des Programmes zur Folge, so daß man auf diese Möglichkeit nur zurückgreifen sollte, wenn man sich sicher ist, daß eine andere Lösung nicht möglich oder nicht vertretbar ist.

Für „Insider" sind in Tabelle 4.1 die Werte der wichtigsten Konstanten von METAFONT angegeben, so wie sie in dieser Implementation gewählt wurden. Eine ausführliche Erklärung dieser Variablen findet man in [2].

main memory size	*mem_max = mem_top*	65 534
	mem_min = mem_bot	0
number of internals	*max_internal*	100
buffer size	*buf_size*	500
error messages	*error_line*	72
	half_error_line	42
text output width	*max_print_line*	79
graphics screen size	*screen_width*	640
	screen_depth	399
input stack	*stack_size*	30
number of strings	*max_strings*	2 000
string pool size	*pool_size*	32 000
	string_vacancies	8 000
GF file buffer	*gf_buf_size*	800
file names	*file_name_size*	250
TFM header words	*header_size*	100
ligature/kern steps	*lig_table_size*	300
fontdimen parameters	*max_font_dimen*	50
symbolic tokens	*hash_size*	2 100
input files	*max_in_open*	6
macro parameters	*param_size*	150

Tabelle 4.1: METAFONT-Konstanten

Kapitel 5

INIMF

Auf der Programmdiskette befinden sich zwei Versionen von METAFONT: METAFONT.PRG und INIMF.PRG. Dabei ist ersteres für die tägliche Arbeit gedacht; INIMF wird im allgemeinen nur zur Erzeugung von Base-Dateien (.BSE) benötigt. Wer bereits die Format-Dateien (.FMT) von TEX kennt, kann den nächsten Absatz überspringen, denn die Base-Dateien bei METAFONT entsprechen genau den Format-Dateien bei TEX.

Beim Programmstart müssen sehr viele Variablen initialisiert und der sogenannte „String-Pool" (Datei MF_POOL) mit allen vordefinierten Strings eingelesen werden. Außerdem wird fast jeder Benutzer zumindest die Makros aus der Datei PLAIN.MF verwenden, da die sogenannten „primitives" – also die Befehle des „nackten" METAFONT ohne Plain-Makros – so primitiv sind (wie der Name schon sagt), daß man sehr viel tippen müßte, um nur ein paar einfache Sachen auf den Bildschirm zu bringen, geschweige denn einen ganzen Zeichensatz. Die „primitives" könnte man auch als die „Maschinensprache" von METAFONT bezeichnen. Wenn METAFONT bei jedem Programmstart erst diese ganzen Initialisierungen ausführen und Makrodefinitionen einlesen müßte, würde das recht lange dauern. Darum hat Knuth eine Möglichkeit vorgesehen, diese Initialisierungen und Definitionen nach einmaliger Ausführung in kompakter Form auf Diskette bzw. Platte abzuspeichern. Diese Datei wird Base-Datei genannt. Wenn man das einmal erledigt hat, braucht das Programm in Zukunft nur noch die Base-Datei einzulesen, und ist danach sofort startbereit.

Der Unterschied zwischen INIMF und METAFONT besteht im wesentlichen darin, daß bei METAFONT die komplette Variablen-Initialisierung und das Lesen des String-Pools entfernt wurde, wodurch es nicht nur schneller, sondern auch um einiges kürzer als INIMF ist. Aus diesem Grund braucht es aber unbedingt eine Base-Datei, um arbeiten zu können. Und zur Erzeugung so einer Base-Datei braucht man eben INIMF, auch deshalb, weil der Befehl zum Schreiben der Base-Datei (dump) im normalen METAFONT gar nicht vorhanden ist.

Ein weiterer Unterschied besteht in der Berechnung einiger „statistischer" Werte während der Arbeit. Wenn man z.B. die Variable *tracingstats* auf einen positiven Wert setzt, so erhält man am Ende der LOG-Datei einige Informationen darüber, wie stark der interne Speicher ausgenutzt wurde. Das Mitführen und Aktualisieren solcher Zusatzinformationen kostet jedoch Zeit und Speicherplatz, deshalb wurden die entsprechenden Code-Teile in `METAFONT.PRG` entfernt. Wenn man diese Informationen benötigt, muß man auf INIMF zurückgreifen, muß dafür aber etwas mehr Zeit und Speicher einkalkulieren. An dieser Stelle sei nochmal betont, daß man im Prinzip auch mit INIMF allein auskäme; alles, was man mit METAFONT machen kann, funktioniert auch mit INIMF (vorausgesetzt, man hat genügend Speicher zur Verfügung). Insbesondere kann man auch bei INIMF eine Base-Datei laden (z.B. durch Eingabe von '`&plain`' in der Startzeile). Lediglich wenn man keine Base-Datei in der Startzeile angibt, lädt METAFONT die in der Setup-Datei angegebene `defaultbase`, während INIMF in diesem Fall gar keine Base-Datei lädt, sondern nur seine Variablen initialisiert und den String-Pool liest (nicht aber die Plain-Makros!).

Abschließend nochmal eine kurze Zusammenfassung der Besonderheiten von INIMF:

- alle Variablen werden beim Programmstart initialisiert;
- der String-Pool wird gelesen (der Dateiname steht in der Setup-Datei unter `poolfile`);
- der Befehl dump erzeugt eine Base-Datei;
- wenn in der Startzeile keine Base-Datei angegeben wird, wird auch keine geladen;

- es werden zusätzliche Statistiken geführt. Diese werden ausgegeben, wenn *tracingstats* > 0 bzw. *tracingedges* > 1;
- es wird mehr Hauptspeicher benötigt (für zusätzlichen Programmcode und die statistischen Informationen);
- es wird mehr Rechenzeit „verbraten“.

Kapitel 6

Erste Schritte mit METAFONT

„Aller Anfang ist schwer“, noch dazu, wenn man so komplexe und flexible Programme wie TEX oder METAFONT vor sich hat. Um dem Anfänger dennoch einen ganz kleinen Ausschnitt der Möglichkeiten von METAFONT zu zeigen und ihm gleich am Anfang ein paar kleine Erfolgserlebnisse zu ermöglichen, sind in diesem Kapitel ein paar Beispiele aufgeführt, die am besten gleich am Rechner ausprobiert werden sollten. Für die ersten beiden Beispiele kann METAFONT direkt von der Programmdiskette aus gestartet werden, das dritte Beispiel ist am einfachsten mit einem fertig installierten METAFONT durchzuführen. Bevor ich's vergesse: beenden kann man das Programm durch Eingabe von 'end' oder 'bye', genauso wie TEX also (wenn man mal davon absieht, daß man in METAFONT kein '\' vor die Befehle setzen muß).

6.1 Beispiel 1: einfache Grafik

Im ersten Beispiel sollen ein paar Pinselstriche auf den Bildschirm gebracht werden. Zunächst muß METAFONT gestartet werden, z.B. durch Anklicken von METAFONT.PRG auf dem Desktop (keine Angst, ich erkläre jetzt nicht, wie man mit GEM umgeht). Nach der üblichen Titelzeile meldet sich das Programm mit zwei Sternchen. Normalerweise kann man hier angeben, welchen

Zeichensatz man erzeugen möchte, und noch viele andere Dinge. Für dieses Beispiel jedoch kann sich METAFONT „ganz entspannt zurücklehnen“ und auf die Eingabe warten, also tippt man

```
\ relax
```

ein. Zunächst muß METAFONT mitgeteilt werden, daß man die gezeichneten Striche gleich auf dem Bildschirm sehen möchte. Dies erreicht man durch Eingabe von

```
screenstrokes;
```

(den Strichpunkt nicht vergessen!). Zum Zeichnen einer Linie ist die Angabe von zwei Punkten nötig, wobei die Koordinaten eines Punktes als `(x,y)` eingegeben werden können:

```
draw (10,10)..(50,30);
```

Wenn man mehr als zwei Punkte angibt, wird eine Kurve durch alle Punkte gezogen (sogenannte Splines), z.B.

```
draw (10, 100)..(50,80)..(90,100);
```

Das ist natürlich nur die einfachste Methode, zwei oder mehr Punkte mit einer Kurve zu verbinden. METAFONT kann noch sehr viel mehr, z.B. kann man für bestimmte Punkte der Kurve eine Richtung vorgeben, mit „Kontrollpunkten“ die Form der Kurve fast beliebig verändern, Schnittpunkte mit anderen Kurven definieren usw. Experimentieren kann man auch mit speziellen Verbindungen, indem man die zwei Punkte '..' durch '...' oder '--' ersetzt. Um das Grafikfenster zu löschen, kann man

```
clearit; showit;
```

eingeben. `clearit` löscht dabei den Bildpuffer, `showit` bringt den gelöschten Puffer dann auf den Bildschirm.

6.2 Beispiel 2: METAFONT als „Rechner“

Im zweiten Beispiel sollen die mathematischen Fähigkeiten von METAFONT etwas ausprobiert werden. Zunächst kann man in METAFONT (wie in fast jeder anderen Programmiersprache auch) Variablen definieren. Man hätte das erste Beispiel auch so eingeben können:

```
x1 = 10;  y1 = 10;  x2 = 50;  y2 = 30;
draw (x1, y1)..(x2, y2);
```

Dabei steht das Gleichheitszeichen nicht für eine Zuweisung, sondern für eine Gleichung. Der Unterschied wird klar, wenn man z.B. den Wert von $x1$ um eins erhöhen will. Schreibt man

```
x1 = x1 + 1;
```

so beschwert sich METAFONT zu Recht, daß diese Gleichung „inkonsistent“ (d.h. nicht lösbar) ist, denn es gibt nun mal keine reelle Zahl, deren Wert bei Addition von eins unverändert bleibt. Will man eine Zuweisung erreichen, muß man ':=' eingeben. Richtig wäre also in diesem Fall:

```
x1 := x1 + 1;
```

Interessanter ist aber wohl die Verwendung von Gleichungen. Lineare Gleichungssysteme sind für METAFONT was ganz alltägliches. Man hätte die obigen Werte also auch so definieren können:

```
 x1 +  x2 = 60;
3x1 + 2x2 = 130;
```

Man kann leicht nachprüfen, daß METAFONT dieses Gleichungssystem korrekt gelöst hat, wenn man

```
show x1, x2;
```

eingibt. Interessant ist auch, daß man nicht '3*x1' schreiben muß, sondern den Malpunkt einfach weglassen kann, so wie es wohl (nicht nur) jeder Mathematiker gewohnt ist. Ich kenne keine andere Programmiersprache, in der solche Selbstverständlichkeiten möglich sind. Es bedarf wohl keiner besonderen Erwähnung, daß für METAFONT auch die Rechenregel „Punkt vor Strich" nichts Unbekanntes ist. Ansonsten bietet METAFONT unter anderem noch trigonometrische Funktionen, approximative (d.h. näherungsweise) Lösung von nichtlinearen Gleichungssystemen, Vektor- und Matrizenrechnung. Durch die Möglichkeit, Makros zu definieren (was im Prinzip den Unterprogrammen oder Prozeduren herkömmlicher Programmiersprachen entspricht), kann man natürlich noch beliebig viele andere mathematische Rechenverfahren implementieren.

6.3 Beispiel 3: Erzeugung eines Zeichensatzes

Zum Schluß dieses Kapitels soll noch erklärt werden, wie man mit METAFONT komplette „Computer Modern Roman" Zeichensätze erzeugen kann. Das ist wahrscheinlich eine der häufigsten Aufgaben dieses Programmes, schließlich wollen (und können) nicht alle Benutzer ihre eigenen Zeichen entwerfen, darum greift man in der Regel auf die fertigen Zeichensätze von Knuth zurück, die ja auch eine lange Entwicklungszeit hinter sich haben und deshalb wenigstens halbwegs professionell aussehen (dennoch hat Knuth nie behauptet, ein Typograph zu sein, er hat sich aber bei der Entwicklung der Zeichensätze von einigen Experten dieses Faches beraten lassen).

Als erstes sollte man nachschauen, ob in der Datei ATARI.MF ein 'mode_def' für den verwendeten Drucker vorhanden ist und wie dieser heißt. Wenn dort der gewünschte Drucker nicht angegeben ist, sollte man nach einem 'mode_def' suchen, dessen Auflösung der gewünschten möglichst nahe kommt oder gleich ist. In diesem Fall wird man wohl früher oder später nicht drumherum kommen, sich einen eigenen Parametersatz zu definieren. Vorher sollte man aber bei mir oder Stefan Lindner anrufen, ob vielleicht inzwischen schon fertige Parameter existieren, es kommen fast täglich neue hinzu.

Als Beispiel soll hier der Zeichensatz 'cmr10' in 1,44-facher Vergrößerung für einen Laserdrucker erzeugt werden, sodaß er in einem TEX-Dokument mit

```
\font\bigtenrm = cmr10 scaled \magstep2
```

angesprochen werden kann.

Erzeugen der Base-Datei

Zunächst muß man dafür sorgen, daß eine geeignete Base-Datei vorhanden ist. Für dieses Beispiel genügen die Plain-Makros, also braucht man die Datei PLAIN.BSE. Es ist zwar auf der Programmdiskette schon eine fertige Datei vorhanden, dennoch soll hier kurz erläutert werden, wie diese Datei erzeugt wurde (schließlich fallen solche Dateien ja nur äußerst selten vom Himmel: das letzte Mal fand so ein bemerkenswertes Ereignis glaube ich im Jahre 1938 statt).

Wie in Kapitel 5 bereits erwähnt wurde, benötigt man für diese Aufgabe das Programm INIMF. Man startet also INIMF.PRG und gibt in der Startzeile (durch '**' gekennzeichnet)

```
plain
```

ein. Daraufhin werden die Plain-Makros geladen und im Speicher abgelegt. Wenn wieder ein '*' erscheint, ist METAFONT zu weiteren Untaten bereit. Jetzt kann man die eigenen Dateien laden, die damit bei jedem Start von METAFONT automatisch mitgeladen werden. Für den Anfang also mal

```
input atari
```

um einige Werte wie Auflösung, Strichstärke etc. für ein paar übliche Ausgabegeräte (z.B. stscreen, stlaser und starnl) zu definieren. Für den Anfang genügt das mal, man kann INIMF mitteilen, daß es nun seinen Speicher „dumpen" soll:

```
dump
```

Zur Belohnung erhält man ein paar neue Dateien:

PLAIN.BSE in dem Ordner, der in der Setup-Datei unter dumppath angegeben wurde. Das ist die Base-Datei, in der alle wichtigen Daten für METAFONT.PRG enthalten sind.

PLAIN.LOG im logpath-Ordner. In diese Datei hat INIMF die ganzen Bildschirm-Ausgaben und noch einiges mehr mitprotokolliert, damit man sich auch später noch ansehen kann, was das Programm alles angestellt hat.

Die Startzeile

Diese Base-Datei kann man nun mit jedem der beiden Programme (INIMF und METAFONT) ruckzuck einladen, indem man in die Startzeile

```
&plain
```

(man beachte das '&'!) eintippt. Dadurch wird die Datei PLAIN.BSE geladen, und man hat alles wieder so, wie es vor dem Eintippen von dump war.

Jetzt wird es aber doch allerhöchste Zeit, daß genau erklärt wird, was die Startzeile ist, und welche Besonderheiten sie aufweist. Die Startzeile ist die erste Zeile, die man für METAFONT eingibt. Eine Möglichkeit besteht bereits in der Kommandozeile *nach* den Optionen. Wenn dort nichts eingegeben wurde, meldet sich das Programm mit zwei Sternchen ('**') am Zeilenanfang, um anzudeuten, daß es die Startzeile erwartet (in den normalen Zeilen steht nur *ein* Sternchen am Zeilenanfang). In dieser Startzeile muß nun angegeben werden, welche Base-Datei geladen werden soll (bei INIMF optional), außerdem leitet METAFONT aus dieser Zeile den „Jobname" ab, nach dem sämtliche Ausgabedateien (GF-, TFM- und LOG-Datei) benannt werden. Die Besonderheiten der Startzeile sind:

- Eine Base-Datei läßt sich durch Voranstellen eines '&' vor dem Dateinamen laden (z.B. '&plain' lädt die Base-Datei 'PLAIN.BSE'). Wenn eine Base-Datei angegeben ist, muß diese auf jeden Fall am Anfang der Startzeile stehen.

- Eingabedateien (*.MF) kann man einfach durch Angabe des Dateinamens laden, während man in den normalen METAFONT-Eingabezeilen 'input *name*' schreiben müsste (z.B. 'cmr10' lädt die Eingabedatei 'CMR10.MF').

- Wenn man in dieser Zeile andere Sachen eingeben möchte, z.B. Initialisierungen irgendwelcher Variablen, bevor eine Datei gelesen wird, muß man mit einem '\' auf den normalen Eingabemodus umschalten. Danach kann man alles machen, was man normalerweise (in '*.MF'-Eingabedateien oder in Zeilen mit nur einem '*' am Anfang) machen kann. Üblicherweise gibt man hier das Ausgabegerät und eventuelle Vergrößerungen an, z.B.

  ```
  \ mode=stlaser; mag=1.44;
  ```

  setzt Auflösung etc. auf die Werte, die im 'mode_def stlaser' in der Datei ATARI.MF angegeben wurden, und vergrößert den Zeichensatz auf 1.44-fache Größe. Will man mit diesen Werten einen Zeichensatz erzeugen, so müssen die Zuweisungen natürlich *vor* dem Laden der Sourcedatei ausgeführt werden (es nützt nichts, wenn METAFONT zuerst den ganzen Zeichensatz erzeugt und erst hinterher erfährt, daß dieser vergrößert werden sollte). Da man nach Eingabe des \ nicht mehr im „Startzeilen-Modus“ ist, muß man die Sourcedatei nun (wie üblich) mit input laden.

- Anhand der Startzeile bestimmt METAFONT den Jobnamen. Dabei richtet sich METAFONT nach der ersten Datei, die eingelesen wird (ohne Beachtung einer eventuell angegebenen Base-Datei), also entweder der erste Dateiname ohne einem '&' davor, oder (falls vor dem '\' keine MF-Datei eingelesen wird) die erste Datei, die nach einem '\' mit input eingelesen wird. Wenn mit den Kommandos in dieser Zeile überhaupt keine MF-Datei eingelesen wird, dann setzt METAFONT den Jobname auf 'MFPUT'. Wenn man also

```
&plain f1 \ input f2
input f3
```

eingibt, heißen die Ausgabedateien ‘F1.xxxGF’, ‘F1.LOG’ und ‘F1.TFM’, weil ‘F1.MF’ die erste Datei ist, die gelesen wird (die Base-Datei wird nicht mitgerechnet, sonst hieße ja alles ‘PLAIN.xxx’).

Wenn man in der ersten Zeile kein Ausgabegerät spezifiziert, nimmt METAFONT ‘mode = proof;’, was zwar zum Ausprobieren ganz nett aussieht (eigentlich sollte sich dieses Schauspiel niemand entgehen lassen, man sieht die Zeichen mal richtig groß und in hervorragender Schärfe auf dem Monitor), aber für einen Gerätetreiber in der Regel nicht so ideal ist (schon allein wegen der Auflösung von 2601.72 dpi). Also muß man das gewünschte Gerät explizit angeben, indem man in der ersten Zeile vor dem Namen der Eingabedatei ‘mode=stlaser;’ angibt. Damit nun aber METAFONT nicht nach der Eingabedatei ‘mode’ sucht, ist vor der Gleichung noch ein ‘\’ nötig. Wenn keine Vergrößerung angegeben wird, ist sie automatisch 1. Allgemein wird die erste Zeile also ungefähr so aussehen:

```
&<Base> \ mode=<Geraet>; mag=<Vergroess.>; input <Datei>;
```

wobei für die Namen in spitzen Klammern entsprechende Werte einzusetzen sind (die Klammern tippt man natürlich nicht mit ein).

Die „Schöpfung“

Jetzt steht alles bereit, um den Zeichensatz zu erzeugen. Man starte also METAFONT.PRG und gebe als Startzeile

```
&plain \ mode=stlaser; mag=1.44; input cmr10
```

ein. Nach einigen Minuten ist METAFONT fertig, und man hat ein paar weitere Dateien auf der Diskette / Platte:

CMR10.432 sollte eigentlich `CMR10.432GF` heißen, aber TOS kann halt leider nur drei Zeichen in der Extension speichern. Die „krumme" Zahl 432 erhält man, wenn man die Geräte-Auflösung von 300 dpi (dots per inch) mit der Vergrößerung (in diesem Fall 1.44) multipliziert. Das ist also nun die GF-Datei, in der die Beschreibung der vielen Zeichen drin ist. Wenn man einen Zeichensatz für ein anderes Gerät generiert hat, steht natürlich nicht 432, sondern die gewählte Auflösung (Vergrößerung nicht zu vergessen) in der Extension, z.B. `CMR10.96G` ('`.96GF`' auf 3 Zeichen verkürzt!) für eine GF-Datei mit 96 dpi und Vergrößerung 1.

CMR10.LOG auch hier hat METAFONT wieder fein säuberlich mitprotokolliert, was es alles gemacht hat.

CMR10.TFM diese Datei benötigt TEX, um zu erfahren, wie groß die einzelnen Zeichen sind (wie sie tatsächlich aussehen, interessiert TEX gar nicht, das geht einzig und allein den Druckertreiber etwas an). TFM steht für „TEX Font Metric File".

Da die meisten Gerätetreiber nur oder zumindest am liebsten sogenannte PK-Files („Packed Font Files") lesen, muß man nun noch die GF-Datei in eine PK-Datei umwandeln (man könnte auch komprimieren sagen). Man startet also `GFTOPK.TTP` und gibt diesem den Namen der GF-Datei und der gewünschten PK-Datei, also z.B.

```
cmr10.432 \prtfonts\res300.slm\mag____1.440\cmr10.pk
```

als Kommandozeile mit auf den Weg (zum Zwecke der Demonstration sei hier mal davon ausgegangen, daß die GF-Datei im aktuellen Ordner und die Zeichensätze für die Druckertreiber im Ordner `\prtfonts` stehen, wobei letzterer entsprechend den Konventionen von Stefan Lindner's Druckertreiber-Familie aufgebaut ist). Damit ist der Zeichensatz fertig und steht zum Ausdrucken bzw. Betrachten auf dem Bildschirm bereit.

Kapitel 7

Zukunftsmusik

Die Arbeit an METAFONT ist noch lange nicht abgeschlossen, da gibt es noch viel zu viele Ideen, was man besser und schöner machen könnte, und sicher sind in der Benutzeroberfläche auch noch ein paar kleine Fehler versteckt (so perfekt wie Knuth bin ich leider immer noch nicht...).

Eine Sache, die mir noch ziemlich schwer im Magen liegt, ist die Erkennung und Behandlung von Lese- und Schreibfehlern beim Zugriff auf Dateien. Da prüft METAFONT bis jetzt noch fast gar nichts ab, was besonders bei Schreibfehlern (z.B. Diskette voll!) ärgerlich ist, weil man dadurch eventuell unvollständige Dateien erhält, ohne daß das Programm einen Ton sagt. Lesefehler sind meist nicht so kritisch, weil die Syntaxprüfung sowas normalerweise sofort merkt. Aber wie so oft habe ich natürlich auch für diese Schwäche des Programmes eine Ausrede: In der Testversion des Compilers, mit der dieses Programm erzeugt wurde, befanden sich noch einige Fehler; unter anderem der, daß die Schreibfunktion keine Schreibfehler weitermeldet, sondern so tut, als wäre alles in Ordnung. Stefan Lindner hat diesen Fehler mit sehr viel Aufwand in seinem TEX „umschifft“, dafür kann er dann eines Tages, wenn die Bibliotheksfunktionen korrekt arbeiten, einen großen Teil der Arbeit in den Papierkorb schmeißen. Bei diesen Aussichten habe ich mich lieber auf andere Aufgaben gestürzt, die auch in einem Jahr noch gebraucht werden. Es empfiehlt sich also, bereits *vor* dem Aufruf von METAFONT nachzuschauen, ob noch genügend Speicherplatz auf der Diskette oder Festplatte frei ist. Ich weiß, wie unangenehm diese Situation besonders für

Diskettenbenutzer ist (obwohl auch viele Festplattenbesitzer ständig über Platzmangel klagen), aber wenn ich erst alles korrigieren wollte, was mich stört oder mir noch nicht so gut gefällt, wäre diese Version wahrscheinlich erst in zwei Jahren verschickt worden, und dann hätte es bestimmt noch längere Gesichter gegeben...

Nun aber zu erfreulicheren Dingen. Ganz begeistert bin ich von der Idee, für die Eingabedateien (ich denke da speziell an den fast 1 MByte großen Sourcecode der Zeichensätze) auch gepackte Formate zuzulassen. Das praktischste wäre momentan wohl das ARC-Format, aber da ist es ziemlich schwer, wenn nicht gar unmöglich, an eine genaue Beschreibung des Formates, geschweige denn an fertige Dekodier-Routinen dranzukommen. Seit kurzem kursiert jedoch auch mindestens eine für den Atari ST angepaßte Version von ZOO, und da hat es der Autor sogar zur Pflicht gemacht, daß der Sourcecode mitverbreitet wird (meiner Meinung nach eine sehr löbliche Idee). Außerdem sind bereits fertige Routinen dabei, um komprimierte Dateien zu expandieren, und das schreit geradezu nach einer Verwendung in anderen Programmen, speziell natürlich in METAFONT. Damit wäre es möglich, den kompletten Sourcecode der Zeichensätze auf einer Diskette unterzubringen, was wohl nicht zuletzt diejenigen Anwender freuen dürfte, die keine Festplatte besitzen.

Momentan ist die Auflösung des Grafikpuffers fest auf 640 mal 400 Punkte eingestellt. Das ist gerade für stolze Besitzer von Ganzseiten-Bildschirmen unpraktisch, weil das Grafik-Fenster dort nur einen kleinen Teil des Bildschirms füllen kann. Die Ausgabe ist aber auf jedem mir bekannten Monitor möglich, einschließlich Farb- und Großmonitore.

Daß bei der Texteingabe kein Cursor zu sehen ist, stört manchmal schon gewaltig. Ich denke, das sollte mit GEM kein allzu großes Problem darstellen, und werde das in naher Zukunft auch ausprobieren.

Schön wäre es auch, wenn man die Setup-Datei von METAFONT aus ändern könnte. Ich stelle mir da einen Eintrag in der Menüleiste vor, ähnlich wie bei vielen anderen GEM-Programmen, wo man die Standard-Pfade angeben und anschließend abspeichern kann. Bei dieser Gelegenheit könnte man auch gleich Größe und Position der Fenster mit abspeichern.

Dies führt zu der allgemeinen Frage, wie sinnvoll eine Menüleiste bei METAFONT wäre. Mir sind bis jetzt noch nicht allzu viele Menüeinträge eingefallen,

die eine Menüleiste rechtfertigen würden. Ich hoffe, daß die Benutzer meiner Phantasie in diesem Punkt etwas nachhelfen. Also, wer konkrete Wünsche hat, was in der Menüleiste (oder vielleicht auch in Icons, Dialogboxen etc.?) angeboten werden soll, melde sich bitte bei mir!

Ein sicher sehr interessantes, aber auch aufwendiges Projekt wäre, METAFONT mit der hervorragenden Benutzeroberfläche von SMALLTALK-80 zu verbinden. Ein erster Ansatzpunkt dazu ist die Möglichkeit, C-Funktionen als „primitive" Methoden zu SMALLTALK-80 dazulinken zu können (und im Prinzip ist METAFONT nichts anderes als eine C-Funktion namens `main`). Benutzern mit weniger als 4 MByte RAM würde das allerdings vermutlich nicht sehr viel bringen (höchstens den Wunsch nach mehr Speicher).

Literaturverzeichnis

[1] Knuth, Donald E. *The METAFONTbook*, Computers and Typesetting Vol. C, Addison-Wesley, Reading (Massachussets) 1986. 361 Seiten.

Die definitive Anleitung zu METAFONT. Auf dieses Buch kann wohl nur verzichten, wer ausschließlich „Computer Modern Roman"-Zeichensätze für verschiedene Ausgabegeräte erzeugt. Wer mit den Parametern etwas experimentieren will oder eigene Grafiken oder gar Zeichensätze erzeugen will, sollte sich das METAFONTbook zulegen, denn da steht praktisch alles drin, was der Benutzer wissen muß.

[2] Knuth, Donald E. *METAFONT: The Program*, Computers and Typesetting Vol. D, Addison-Wesley, Reading (Massachussets) 1986. 560 Seiten.

Wer es ganz genau wissen will, kann hier den Original-Sourcecode in WEB nachlesen. Dieses Buch ist die Grundlage meiner C-Version von METAFONT.

[3] Knuth, Donald E. *TEX: The Program*, Computers and Typesetting Vol. B, Addison-Wesley, Reading (Massachussets) 1984. 594 Seiten.

Der Sourcecode von TEX für alle, die auch dort hinter die Kulissen schauen wollen. Ebenfalls in der Programmier- und Dokumentationssprache WEB geschrieben.

[4] Knuth, Donald E. *Computer Modern Typefaces*, Computers and Typesetting Vol. E, Addison-Wesley, Reading (Massachussets) 1986. 590 Seiten.

Hier sind die ganzen METAFONT-Programme für die komplette Computer Modern Schriftenfamilie dokumentiert, zusammen mit vielen Probeausdrücken (proofs). Man kann dieses Buch also als Nachschlagewerk,

als Bilderbuch oder auch als Lehrbuch für beispielhafte METAFONT-Programmierung ansehen.

[5] Knuth, Donald E. *The WEB System of Structured Documentation*, Stanford Computer Science Report No. 980, Stanford University, Department of Computer Science, Stanford (California) 1983. 206 Seiten.

Anleitung zu WEB und Sourcecode zu WEAVE und TANGLE, die zusammen das WEB-System bilden.